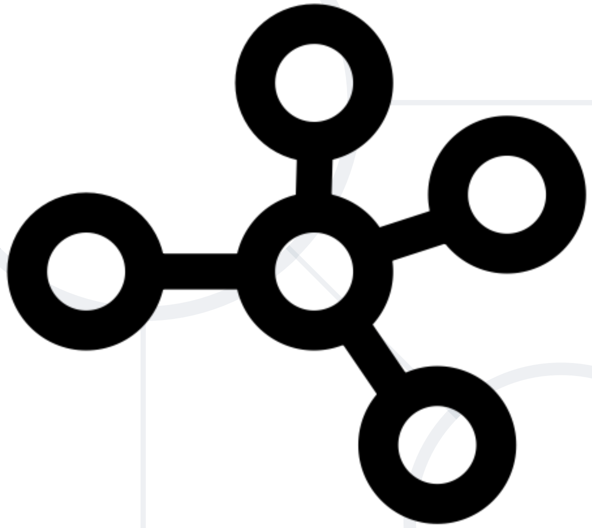
# Dependency Injection



SoftUni Team

Technical Trainers

Software University

http://softuni.bg

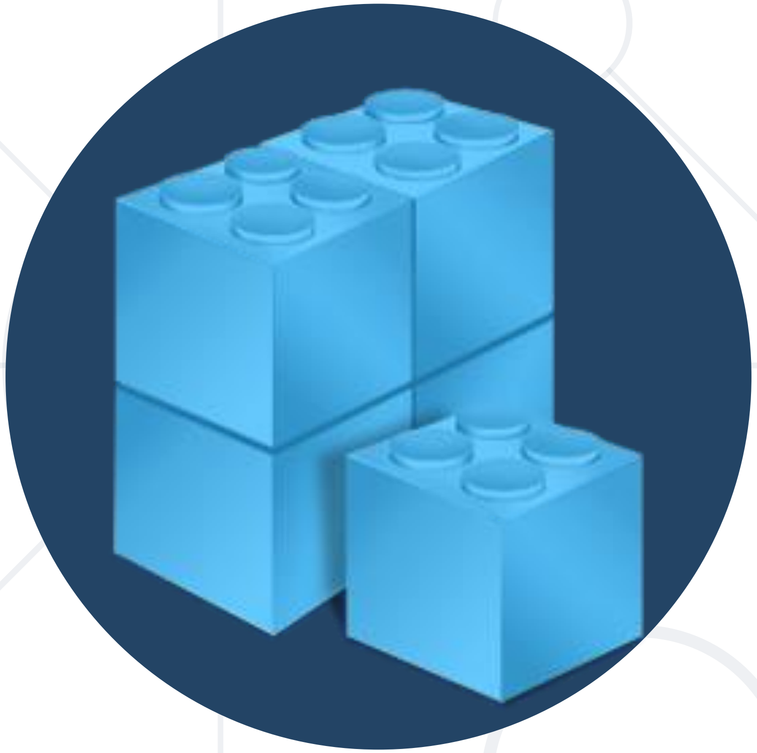
## Have a Question?

Software University

sli.do

# #csharp-advanced

## Table of Contents



# Dependency Injection Overview

A design pattern in programming

## What is a Dependency?

- Another object that your class needs
  - Other Examples (Framework, Database, File System, Providers)
- Classes dependent on each other are called coupled
- Dependencies are bad because they decrease reuse

```
public class Customer
{
  private CustomerService customerService;
  public Customer()
  {
    this.customerService = new CustomerService();
  }
}
```

## Dependency Injection

- Dependency Injection is a popular design pattern
- Inversion of Control (IoC)
  - Dependencies are pushed in the class from the outside
  - The class does not instantiate it's dependencies

```
public class Customer
{
  private customerService;
  public Customer(CustomerService customerService)
  {
    this.customerService = customerService;
  }
}
```

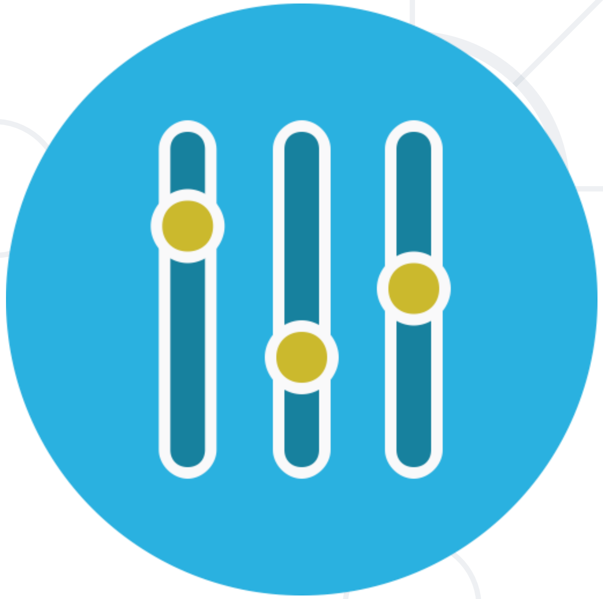
## Dependency Injection

- How it should be
  - Classes should declare what they need
  - Constructors should require dependencies
  - Dependencies should be abstractions
- How to do it
  - Dependency Injection (usually called DI)
  - The Hollywood principle
    "Don't call us, we'll call you!"

# Types of Dependency Injection

Constructor injection

Property injection



Parameter injection

# Constructor Injection – Pros and Cons

- Pros
  - Class' requirements are self-documenting
  - We don't have to worry about state validation
- Cons
  - Too many parameters
  - Sometimes, the functionality doesn't need all of the dependencies

## Constructor Injection - Example

```
class Copy
{
    private IReader reader;
    private IWriter writer;

    public Copy(IReader reader, IWriter writer)
    {
        this.reader = reader;
        this.writer = writer;
    }

    // Read/Write data through the reader/writer
}

var copy = new Copy(new ConsoleReader(),
                    new FileWriter("out.txt"));
```

## Property Injection – Pros and Cons

- Pros
  - Functionality can be changed at any time
  - That makes the code very flexible
- Cons
  - State can be invalid
  - Less intuitive to use

## Property Injection - Example

```
class Copy
{
    public IReader Reader { get; set; }
    public IWriter Writer { get; set; }

    public void CopyAllChars(reader, writer)
    {
        // Read/Write data through the reader/writer
    }
}

Copy copy = new Copy();
copy.Reader = new ConsoleReader();
copy.Writer = new FileWriter("output.txt");
copy.CopyAllChars();
```

## Parameter Injection – Pros and Cons

- Pros
  - Changes are only localized to the method

- Cons
  - Too many parameters
  - Breaks the method signature

## Parameter Injection - Example

```
class Copy
{
  public CopyAllChars(IReader reader, IWriter writer)
  {
    // Read/Write data through the Reader/Writer
  }
}
Copy copy = new Copy();
var reader = new ConsoleReader();
var writer = new FileWriter("output.txt");
copy.CopyAllChars(reader, writer);
```

# DIP Violations

- Classic DIP Violations:
  - Using the **`new`** keyword
  - Using **`static`** methods / properties
- How to fix code, that violates the DIP:
  - **Extract interfaces** + use **constructor injection**
  - Set up an Inversion of Control (**IoC**) container

## Framework Overview

## What is framework?

- A framework is a reusable, "semi-complete" application that can be specialized to produce custom applications.

"Johnson and Foote 1988"

ASP.NET Core

Entity Framework Core

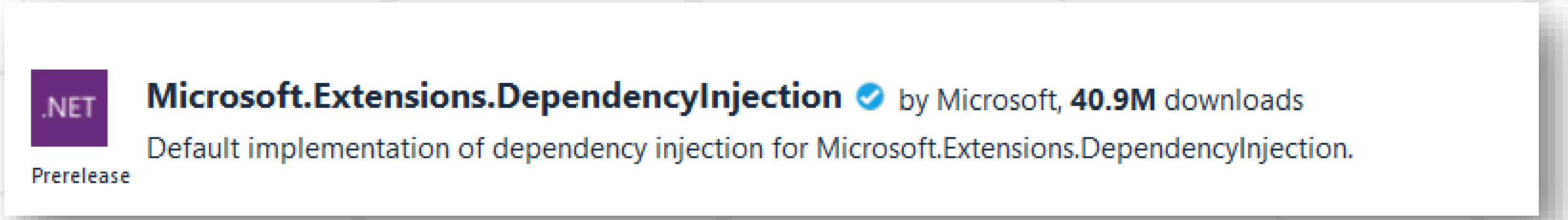
## Framework goals

- Reuse: code, design, analysis and documentation
- Simplify software development
- Reduce code writing
- Allow inexperienced programmers to develop good software
- Extract the knowledge of experienced programmers

## Microsoft Dependency Injection

# Microsoft Dependency Injection

- Install **Microsoft.Extensions.DependecyInjection**



**Microsoft.Extensions.DependencyInjection** by Microsoft, **40.9M** downloads
Default implementation of dependency injection for Microsoft.Extensions.DependencyInjection.
Prerelease

- Define IoC Container

```csharp
private static IServiceProvider ConfigureServices()
{
    var serviceCollection = new ServiceCollection();

    serviceCollection.AddTransient<IHashService, HashService>();
    serviceCollection.AddScoped<IUserService, UserService>();
    serviceCollection.AddSingleton<IUserSessionService, UserSessionService>();

    var serviceProvider = serviceCollection.BuildServiceProvider();

    return serviceProvider;
}
```

## Register Services

- **AddTransient<Interface, Implementation>()**
  - New instance is provided to every controller and every service
- **AddScoped<Interface, Implementation>()**
  - Objects are the same within a request, but different across different requests
- **AddSingelton<Interface, Implementation>()**
  - Only one instance is provided

## Custom DI Framework

Live Demo

## Summary

- Dependency Injection provides better code quality
- Testable
- Maintainable
- Reusable
- Readable
- Implementing Custom Framework

# Questions?

Software University

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

# SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

top EMPLOYER България Bulgaria 2024 FOR A BETTER WORLD OF WORK

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

# License

# Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity